LAPORAN PRAKTIKUM 6
PEMROGRAMAN LANJUT
BUKU ABSENSI MAHASISWA

Oleh

Nama	:	Rizqillah
NIM	:	1957301020
Kelas	:	TI 2C
Dosen Pembimbing	:	Musta'inul Abdi, SST., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI DAN KOMPUTER

TAHUN 2021

LEMBAR PENGESAHAN

No. Praktikum	:	06/PPL/2C/TI/2021
Judul	:	Buku Absensi Mahasiswa
Nama	:	Rizqillah
NIM / Kelas	:	1957301020 / TI 2C
Jurusan	:	Teknologi Informasi Dan Komputer
Prodi	:	Teknik Informatika
Tanggal praktikum	:	9 April 2021
Tanggal penyerahan	:	11 April 2021
Nilai	:	

Buketrata, 11 April 2021

Dosen Pembimbing,

Musta'inul Abdi, SST., M.Kom.
NIP. 19911030 20190310 1 5

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Tujuan

Modul ini mengenalkan suatu teknik pemrograman yang lebih tinggi. Dalam bagian ini Anda akan mempelajari Java Collections.

Setelah menyelesaikan pelajaran ini, diharapkan Anda dapat:

1. Memahami dan menggunakan Java Collections
2. Membuat dan mengelola Java Collections dengan baik
3. Memahami dan menggunakan OOP pada program Java

1.2 Object Oriented Programming

OOP (Object Oriented Programming) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Setiap bagian dari suatu permasalahan adalah objek, objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Contohnya seperti sebuah Sepeda Motor, pada sepeda motor pasti memiliki bagian-bagian dari yang terbesar sampai bagian terkecil. Dan bagian-bagian inilah yang disebut objek yang memiliki class-class nya masing-masing. Dan setiap class-class tersebut bisa saja memiliki method yang banyak ataupun method yang sedikit. Setiap method itu bisa digunakan agar melakukan pekerjaan yang diinginkan, yaitu seperti jika ingin membuat method pada lampu sepeda motor agar lampunya hidup, ketika mencoba memanggil method tersebut, maka lampu sepeda motor tersebut akan hidup.

Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain

1.2.1 Class

Class adalah suatu blueprint dari sebuah hal yang akan dicetak, yaitu objek. Class di dalam Java digunakan untuk membuat sebuah kerangka kerja. Bisa dikatakan sebagai library. Class berisi property dan method. Class adalah sebuah wadah yang menyimpan property dan method. Dan objek adalah yang dihasilkan berdasarkan dari class tersebut.

1.2.2 Objek

Objek adalah hasil cetakan dari sebuah class, atau hasil konkrit dari class. Jika dianalogikan dari class sebuah cetakan kue. Maka kue yang dihasilkan tersebut adalah objek dari hasil blueprintnya cetakan tersebut.

1.2.3 Method

Method atau disebut juga tingkah laku adalah hal-hal yang bisa dilakukan oleh object dari suatu Class. Method dapat digunakan untuk mengubah nilai atribut suatu object, menerima informasi dari object lain, dan mengirim informasi dari object lain. Cara object berkomunikasi dengan object lain adalah dengan menggunakan method.

1.3 Collections

Collections framework merupakan bentuk algoritma yang digunakan untuk merepresentasikan dan memanipulasi collections. Semua collections frameworks mengandung hal-hal berikut:

- **Interfaces:** memungkinkan collections dimanipulasi secara independen.
- **Implementations:** merupakan implementasi dari collection interfaces. Dan merupakan struktur data yang reusable.
- **Algorithms:** merupakan method-method yang dapat digunakan untuk melakukan proses komputasi tertentu, seperti searching (pencarian) dan sorting (pengurutan), terhadap objek yang meng-implement collection interfaces. Method-method pada Java Collections Framework adalah polymorphic: maksudnya nama method yang sama dapat digunakan pada collection interface yang sesuai. Algoritma pada Java Collections Framework memiliki fungsi yang reusable.

Java telah menyajikan Collection class dan interface yang lain, yang semuanya dapat ditemukan di java.util package. Contoh dari Collection classes termasuk LinkedList, ArrayList, HashSet dan TreeSet. Class tersebut benar-benar implementasi dari collection interfaces yang berbeda. Induk hirarki dari collection interfaces adalah collection interfaces itu sendiri. Sebuah collection hanya sebuah grup dari object yang diketahui sebagai elemennya sendiri. Collection memperbolehkan penggandaan/salinan dan tidak membutuhkan pemesanan elemen secara spesifik.

SDK(Software Development Kit) tidak menyediakan implementasi built-in yang lain dari interface ini tetapi mengarahkan subinterfaces, Set interfaces dan List interfaces diperbolehkan. Sekarang, apa perbedaan dari kedua interface tersebut.

Set merupakan collection yang tidak dipesan dan tidak ada penggandaan didalamnya. Sementara itu, list merupakan collection yang dipesan dari elemen-elemen dimana juga diperbolehkannya penggandaan. HashSet, LinkedHashSet dan TreeSet suatu implementasi class dari Set interfaces. ArrayList, LinkedList dan Vector suatu implementasi class dari List interfaces

1.3.1 Interface List

Interface List merupakan sub-interface dari interface Collection. Interface List digunakan untuk mengkoleksi data dalam bentuk terurut dan memperbolehkan duplikasi. Interface List menambahkan operasi yang berkaitan dengan posisi.

Dua class penting yang ada dalam Java Collections Framework yang meng-implement interface List adalah: ArrayList dan LinkedList. Jika perlu List yang dapat menambahkan dan menghapus data dimana saja (random access) maka dapat menggunakan ArrayList. Namun jika perlu List yang dapat menambah dan menghapus data di sekitar tengah-tengah dan mengaksesnya secara sekuensial, maka LinkedList adalah pilihannya.

ArrayList menyimpan data seperti array (diakses dengan index) namun ukurannya dapat bertambah secara fleksibel. Elemen yang dapat dimasukkan dalam ArrayList bisa bermacam-macam, termasuk null. ArrayList bisa disamakan dengan class Vector namun bedanya ArrayList ini unsynchronized. Operasi size, isEmpty, get, set, iterator, dan listIterator berjalan dalam waktu konstan dan lebih cepat dari LinkedList. ArrayList merupakan versi fleksibel dari array biasa. Yang mengimplementasikan List interface.

LinkedList merupakan implementasi dari algoritma LinkedList yang dipelajari dalam struktur data.

Adapun perbedaan LinkedList dan ArrayList sebagai berikut :

1. ArrayList secara internal menggunakan array dinamis untuk meyimpan elemen. Sementara LinkedList secara internal digunakan untuk menyimpan elemen dari list tertaut ganda. Atau bisa dikatakan linkedlist menyimpan elemennya dengan cara menghubungkan elemen sebelumnya ke elemen selanjutnya.
2. Manipulasi dengan ArrayList cenderung lambat karena secara internal menggunakan sebuah array. Jika elemen lainnya dipindahkan atau dihilangkan dari Array, maka akan terjadi pembuatan array yang baru. Sementara pada LinkedList lebih cepat dibanding ArrayList karena menggunakan linked list yang terhubung, sehingga tidak diperlukan sedikit pergeseran dalam memori.

3. Sebuah kelas ArrayList dapat menjadi seperti sebuah list karena hal tersebut hanya mengimplementasi List saja. Sementara kelas LinkedList dapat menjadi sebuah list dan antrian karena mengimplementasikan interface List dan Dequeue.
4. ArrayList lebih baik untuk menyimpan dan mengakses data, sementara LinkedList lebih baik untuk manipulasi data. Dikarenakan Arraylist menggunakan pengalamatan dengan index. Akan tetapi Linkedlist menggunakan penunjuk/pointer, oleh karenanya jika ingin mengakses elemennya, maka akan mengecek elemen dari yang pertama ke selanjutnya.

Adapun dalam keadaan apakah lebih baik menggunakan LinkedList atau ArrayList :

1. Operasi penyisipan dan penghapusan memberikan kinerja yang baik di LinkedList dibandingkan dengan ArrayList. Karenanya jika ada persyaratan untuk sering menambahkan dan menghapus data, maka LinkedList adalah pilihan terbaik.
2. Operasi pencarian (method get) cepat di Arraylist tetapi tidak di LinkedList jadi jika ada lebih sedikit operasi tambah dan hapus dan lebih banyak persyaratan operasi pencarian, ArrayList akan menjadi pilihan terbaik Anda.

1.3.2 Linked List

LinkedList adalah bagian dari Java Collection yang ada dalam paket java.util. Kelas ini merupakan implementasi dari struktur data LinkedList yang merupakan struktur data linier dimana elemen tidak disimpan di lokasi yang berdekatan dan setiap elemen merupakan objek terpisah dengan bagian data dan bagian alamat. Elemen-elemen tersebut dihubungkan menggunakan pointer dan alamat. Setiap elemen dikenal sebagai node. Karena dinamik dan kemudahan penyisipan dan penghapusan, LinkedList lebih disukai daripada Array. Ini juga memiliki beberapa kelemahan seperti node tidak dapat diakses secara langsung sebagai gantinya perlu memulai dari head dan mengikuti link untuk mencapai node yang ingin diakses.

Poin penting tentang Java LinkedList adalah :

- Kelas Java LinkedList dapat berisi elemen duplikat.
- Kelas Java LinkedList mempertahankan urutan penyisipan.
- Kelas Java LinkedList tidak disinkronkan.
- Kelas Java LinkedList, manipulasi cepat karena tidak perlu terjadi pergeseran.
- Kelas Java LinkedList dapat digunakan sebagai list, tumpukan atau antrian.

Adapun Method LinkedList :

1. Menambahkan Elemen : Untuk menambahkan elemen ke LinkedList, bisa menggunakan method add(). Adapun contoh dari method add adalah sebagai berikut ini :
   - add (Object): Method ini digunakan untuk menambahkan elemen di akhir LinkedList.
   - add (int index, Object): Method ini digunakan untuk menambahkan elemen pada indeks tertentu di LinkedList.
2. Mengubah Elemen : Setelah menambahkan elemen, jika ingin mengubah elemen, dapat dilakukan dengan menggunakan method set(). Karena LinkedList tidak memiliki index, maka elemen yang ingin diubah direferensikan oleh pointer antara satu wadah ke wadah lain. Oleh karena itu, method ini mengambil pointer dan elemen yang diperbarui yang perlu disisipkan pada wadah tersebut.
3. Menghapus Elemen: Untuk menghapus elemen dari LinkedList, dapat menggunakan method remove(). Method ini memiliki beberapa macam contoh pemanggilan dan parameter, yaitu sebagai berikut :
   - remove(Objek): Method ini digunakan untuk menghapus objek dari LinkedList. Jika ada beberapa objek seperti itu, kemunculan pertama objek tersebut akan dihapus.
   - remove(int index): Karena LinkedList tidak memiliki index, maka method ini mengambil nilai integer yang hanya menghapus elemen yang ada di wadah tertentu di LinkedList. Setelah menghapus elemen, semua elemen dipindahkan ke kiri untuk mengisi ruang dan wadah objek diperbarui.
4. Iterasi LinkedList : Ada beberapa cara untuk melakukan iterasi melalui LinkedList. Cara yang paling terkenal adalah dengan menggunakan perulangan for dasar yang dikombinasikan dengan method get() untuk mendapatkan elemen pada indeks tertentu dan perulangan for lanjutan.
5. Membersihkan LinkedList : pada LinkedList, method yang bisa digunakan untuk menghapus semua elemen LinkedList adalah dengan method clear().

Dan adapun Method lainnya seperti :

1. addAll(int index, Collection<E> c) **:** Method ini Menyisipkan semua elemen dalam koleksi yang ditentukan ke dalam list, dimulai dari posisi yang ditentukan.
2. addFirst(E e) : Method ini Menyisipkan elemen yang ditentukan di awal list.
3. addLast(E e) : Method ini Menyisipkan elemen yang ditentukan di akhir list.
4. get(int index) : Method ini mengembalikan elemen pada posisi yang ditentukan dalam list.

5. getFirst() : ini mengembalikan elemen pada posisi pertama dalam list.
6. getLast() : ini mengembalikan elemen pada posisi terakhir dalam list.
7. peek() : Method ini akan mengembalikan nilai head dari list. Akan tetapi tidak akan menghapus nilai.
8. pop() : Method ini akan mengembalikan nilai menggunakan fungsi seperti stack, dan nilai tersebut akan dihapus.
9. push() : Method ini akan mengisi data pada LinkedList seperti operasi pada Stack.
10. size() : Method ini akan mengembalikan nilai dari jumlah data dalam list.

1.3.3 ArrayList

ArrayList adalah bagian dari Java Collection dan berada dalam paket java.util. ArrayList memberi array dinamis di Java Collection. Meskipun, ini mungkin lebih lambat dari array standar tetapi dapat membantu dalam program yang membutuhkan banyak manipulasi dalam array.

Kelas Java ArrayList menggunakan array dinamis untuk menyimpan elemen. Ini seperti sebuah array, tetapi tidak ada batasan ukuran. ArrayList dapat menambah atau menghapus elemen kapan saja. Jadi, ini jauh lebih fleksibel daripada array tradisional. Ini dapat ditemukan di paket java.util. ArrayList ini seperti Vektor di C++.

ArrayList di Java juga dapat memiliki elemen duplikat. Ini mengimplementasikan interface list sehingga dapat menggunakan semua method interface list. ArrayList mempertahankan urutan penyisipan secara internal.

Poin Penting dari ArrayList :

- ArrayList mewarisi kelas AbstractList dan mengimplementasikan interface List.
- ArrayList diinisialisasi oleh ukurannya. Namun, ukurannya bertambah secara otomatis jika koleksi bertambah atau menyusut jika objek dikeluarkan dari koleksi.
- Java ArrayList memungkinkan untuk mengakses list secara acak.
- ArrayList tidak dapat digunakan untuk tipe primitif, seperti int, char, dll. Maka akan membutuhkan kelas pembungkus untuk kasus seperti itu.
- ArrayList di Java dapat dilihat sebagai vektor di C ++.
- ArrayList tidak disinkronkan. Kelas tersinkronisasi yang setara di Java adalah Vector.

Bagaimana ArrayList bekerja secara internal ?

Karena ArrayList adalah array dinamis dan tidak perlu menentukan ukurannya saat membuatnya, ukuran array otomatis bertambah saat menambahkan dan menghapus item secara dinamis. Meskipun implementasi library sebenarnya mungkin lebih kompleks, berikut ini adalah ide yang sangat mendasar yang menjelaskan cara kerja array saat array menjadi penuh dan jika mencoba menambahkan item :

- Membuat memori berukuran lebih besar pada memori tumpukan(misalnya memori berukuran ganda).
- Menyalin elemen memori saat ini ke memori baru.
- Item baru yang ditambahkan sekarang karena ada memori yang lebih besar yang tersedia sekarang.
- Hapus memori lama.

Method ArrayList :

1. Menambahkan Elemen: Untuk menambahkan elemen ke ArrayList, bisa menggunakan method add(). Method ini beberapa jenis, yaitu sebagai berikut:
   - add (Object): Method ini digunakan untuk menambahkan elemen di akhir ArrayList.
   - add (int index, Object): Method ini digunakan untuk menambahkan elemen pada indeks tertentu di ArrayList.
2. Mengubah Elemen : Setelah menambahkan elemen, jika ingin mengubah elemen, dapat dilakukan dengan menggunakan method set(). Karena ArrayList memiliki index, maka elemen yang ingin diubah direferensikan oleh index elemen. Oleh karena itu, method ini mengambil index dan elemen yang diperbarui yang perlu disisipkan pada index tersebut.
3. Menghapus Elemen: Untuk menghapus elemen dari ArrayList, dapat menggunakan method remove(). Method ini memiliki beberapa macam contoh pemanggilan dan parameter, yaitu sebagai berikut :
   - remove(Objek): Method ini digunakan untuk menghapus objek dari ArrayList. Jika ada beberapa objek seperti itu, kemunculan pertama objek tersebut akan dihapus.
   - remove(int index): Karena ArrayList memiliki index, maka method ini mengambil nilai integer yang hanya menghapus elemen yang ada di indeks tertentu di ArrayList. Setelah menghapus elemen, semua elemen dipindahkan ke kiri untuk mengisi ruang dan indeks objek diperbarui.
4. Iterasi ArrayList : Ada beberapa cara untuk melakukan iterasi melalui ArrayList. Cara yang paling terkenal adalah dengan menggunakan perulangan for dasar yang dikombinasikan dengan method get() untuk mendapatkan elemen pada indeks tertentu dan perulangan for lanjutan.

5. Membersihkan LinkedList : pada LinkedList, method yang bisa digunakan untuk menghapus semua elemen LinkedList adalah dengan method clear().
6. get(int index) : Method ini mengembalikan elemen pada posisi yang ditentukan dalam list.
7. size() : Method ini akan mengembalikan nilai dari jumlah data dalam list.
8. isEmpty() : Method ini akan mengecek apakah nilai dari ArrayList masih kosong atau tidak. Dan keluarannya adalah true atau false.
9. iterator() : Mengembalikan nilai iterator atas elemen dalam list dalam urutan yang benar.

1.3.4 Interface Set

Interface Set merupakan sub-interface dari interface Collection. Interface Set tidak membolehkan duplikasi data di dalam collection. Method yang ada dalam interface Set sama dengan interface Collection. Method paling penting pada interface Set adalah equals() yang digunakan untuk mengecek kesamaan objek.

Dua class penting yang ada dalam Java Collections Framework yang meng-implement interface Set adalah : HashSet dan TreeSet. HashSet merupakan class yang sering digunakan untuk menyimpan collection yang bebas duplikasi. Untuk efisiensi, objek yang ditambahkan dalam HashSet, perlu untuk menggunakan method hashCode(). TreeSet merupakan class yang sering digunakan untuk mengekstrak elemen dari collection dalam urutan tertentu. Agar TreeSet berjalan dengan baik, elemen yang ditambahkannya ke dalamnya harus dapat diurut. Terkadang lebih mudah untuk menambahkan data ke dalam HashSet baru kemudian dikonversi ke TreeSet agar mudah diurut.

Untuk mengoptimalkan ruang penyimpanan HashSet, maka dapat melakukan tuning initial capacity dan load factor. Class TreeSet tidak memiliki opsi tuning karena tree selalu dalam kondisi seimbang, dan memastikan performa log untuk proses insert, hapus dan query.

Perbedaan HashSet dan TreeSet :

1. **HashSet**
   - Kelas menawarkan kinerja waktu yang konstan untuk operasi dasar (menambah, menghapus, memuat dan ukuran).
   - Tidak menjamin bahwa urutan elemen akan tetap konstan seiring waktu
   - Kinerja iterasi tergantung pada kapasitas awal dan faktor beban HashSet.
     - Cukup aman untuk menerima faktor muatan default, tetapi Anda mungkin ingin menentukan kapasitas awal yang kira-kira dua kali ukuran yang Anda harapkan akan meningkat.

2. **TreeSet**
   - menjamin log (n) biaya waktu untuk operasi dasar (menambah, menghapus, dan memuat)
   - menjamin bahwa elemen himpunan akan diurutkan (naik, alami, atau yang ditentukan melalui konstruktornya)
   - tidak menawarkan parameter penyetelan untuk kinerja iterasi
   - menawarkan beberapa method yang berguna untuk menangani set memerintahkan seperti first(), last(), headSet(), dan tailSet()lain-lain
3. **Poin-poin penting :**
   - Keduanya menjamin koleksi elemen yang bebas duplikat
   - Biasanya lebih cepat untuk menambahkan elemen ke HashSet dan kemudian mengonversi koleksi ke TreeSet untuk traversal yang diurutkan duplikat.
   - Tidak satu pun dari implementasi ini yang disinkronkan. Itu adalah jika beberapa utas mengakses satu set secara bersamaan, dan setidaknya satu utas mengubah set, itu harus disinkronkan secara eksternal.
   - LinkedHashSet dalam beberapa hal menengah antara HashSet dan TreeSet. Diimplementasikan sebagai tabel hash dengan daftar tertaut yang berjalan melewatinya, bagaimanapun, HashSet menyediakan iterasi penyisipan yang tidak sama dengan traversal yang diurutkan yang dijamin oleh TreeSet.

1.3.5 HashSet

Kelas HashSet mengimplementasikan interface Set, didukung oleh tabel hash yang sebenarnya merupakan instance HashMap. Tidak ada jaminan dibuat untuk urutan iterasi dari himpunan yang berarti bahwa kelas tidak menjamin urutan elemen yang konstan dari waktu ke waktu. Kelas ini mengizinkan elemen null. Kelas ini juga menawarkan kinerja waktu yang konstan untuk operasi dasar seperti tambah, hapus, isi, dan ukuran dengan asumsi fungsi hash menyebarkan elemen dengan benar di antara set.

Kerja internal dari HashSet : Semua kelas interface Set didukung secara internal oleh Map. HashSet menggunakan HashMap untuk menyimpan objeknya secara internal. Untuk memasukkan nilai di HashMap memerlukan pasangan nilai kunci, tetapi di HashSet, hanya memberikan satu nilai.

Penyimpanan di HashMap : Sebenarnya nilai yang dimasukkan ke dalam HashSet bertindak sebagai kunci untuk Objek peta dan untuk nilainya, java menggunakan variabel konstan. Jadi dalam key-value pair, semua nilainya akan sama.

Poin Penting dari HashSet :

- Menerapkan Set Interface.
- Struktur data yang mendasari HashSet adalah Hashtable.
- Saat menerapkan Set Interface, nilai duplikat tidak diperbolehkan.
- Objek yang Anda sisipkan di HashSet tidak dijamin akan disisipkan dalam urutan yang sama. Objek disisipkan berdasarkan kode hash mereka.
- Elemen NULL diperbolehkan di HashSet.
- HashSet juga mengimplementasikan interface Serializable dan Cloneable.

Method HashSet :

1. Menambahkan Elemen: Untuk menambahkan elemen ke HashSet, bisa menggunakan method add(). Namun, urutan penyisipan tidak disimpan di HashSet. Jadi perlu mencatat bahwa elemen duplikat tidak diperbolehkan dan semua elemen duplikat diabaikan.
2. Menghapus Elemen : Nilai dapat dihapus dari HashSet menggunakan method remove().
3. Iterasi melalui HashSet : Iterasi melalui elemen HashSet menggunakan method iterator(). Yang paling terkenal adalah menggunakan for loop yang disempurnakan.
4. size() : Method ini akan mengembalikan nilai dari jumlah data dalam set.
5. isEmpty() : Method ini akan mengecek apakah nilai dari HashSet masih kosong atau tidak. Dan keluarannya adalah true atau false.
6. clear : digunakan untuk menghapus semua elemen HashSet.

1.3.6 TreeSet

TreeSet adalah salah satu implementasi terpenting dari interface Set di Java yang menggunakan Tree untuk penyimpanan. Urutan elemen dipertahankan oleh himpunan menggunakan pengurutan alami atau pembanding eksplisit disediakan atau tidak. TreeSet harus konsisten dengan equals() jika ingin mengimplementasikan interface Set dengan benar. Itu juga dapat dilakukan oleh pembanding yang disediakan pada waktu pembuatan TreeSet, tergantung pada konstruktor mana yang digunakan. TreeSet mengimplementasikan interface NavigableSet dengan mewarisi kelas AbstractSet.

Kelas Java TreeSet mengimplementasikan interface Set yang menggunakan tree untuk penyimpanan. Ini mewarisi kelas AbstractSet dan mengimplementasikan interface NavigableSet. Objek dari kelas TreeSet disimpan dalam urutan menaik.

Poin Penting TreeSet :

- Kelas Java TreeSet berisi elemen unik hanya seperti HashSet.
- Akses kelas Java TreeSet dan waktu pengambilan cukup cepat.
- Kelas Java TreeSet tidak mengizinkan elemen null.
- Kelas Java TreeSet tidak tersinkronisasi.
- Kelas Java TreeSet mempertahankan urutan menaik.

Method TreeSet :

1. Menambahkan Elemen : Untuk menambahkan elemen ke TreeSet, dapat menggunakan method add(). Namun, urutan penyisipan tidak dipertahankan di TreeSet. Secara internal, untuk setiap elemen, nilainya dibandingkan dan diurutkan dalam urutan menaik. Perlu dicatat bahwa elemen duplikat tidak diperbolehkan dan semua elemen duplikat diabaikan. Dan juga, nilai Null tidak diterima oleh TreeSet.
2. Mengakses Elemen : Setelah menambahkan elemen, jika ingin mengakses elemen, dapat menggunakan method bawaan seperti contains(), first(), last(), dll.
3. Menghapus Nilai : Nilai dapat dihapus dari TreeSet menggunakan method remove(). Ada berbagai method lain yang digunakan untuk menghapus nilai pertama atau nilai terakhir.
4. Iterasi melalui TreeSet : Ada berbagai cara untuk melakukan iterasi melalui TreeSet. Yang paling terkenal adalah menggunakan for loop yang disempurnakan.

Bagaimana TreeSet bekerja secara internal?

TreeSet pada dasarnya adalah implementasi dari pohon pencarian biner yang menyeimbangkan diri seperti Pohon Merah-Hitam. Oleh karena itu operasi seperti menambah, menghapus, dan mencari membutuhkan waktu O (log (N)). Alasannya adalah pada pohon self-balancing, dipastikan bahwa tinggi pohon selalu O (log (N)) untuk semua operasi. Oleh karena itu, ini dianggap sebagai salah satu struktur data yang paling efisien untuk menyimpan data terurut besar dan melakukan operasi padanya. Namun, operasi seperti mencetak N elemen dalam urutan yang diurutkan membutuhkan waktu O (N).

BAB II
PEMBAHASAN

2.1 Absensi Mahasiswa

Program :

```
import java.util.*;
class absenMahasiswa {
    private String nama, nim;
    private int hadir, noAbsen;

    public void setNama(String nama) {
        this.nama = nama;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public void setHadir(int hadir) {
        this.hadir = hadir;
    }

    public void setNoAbsen(int noAbsen) {
        this.noAbsen = noAbsen;
    }

    public String getNama() {
        return nama;
    }

    public String getNim() {
        return nim;
    }

    public int getHadir() {
        return hadir;
    }

    public int getNoAbsen() {
        return noAbsen;
    }

    public absenMahasiswa(String nama, String nim, int hadir, int
noAbsen) {
        setNama(nama);
        setNim(nim);
        setHadir(hadir);
```

```
        setNoAbsen(noAbsen);
    }

    public void print() {
        String dataHadir;
        if (hadir == 1) {
            dataHadir = "Hadir";
        } else if (hadir == 2) {
            dataHadir = "Sakit";
        } else {
            dataHadir = "Alpa";
        }

System.out.println("=========================================="
);
        System.out.println("\nNo\t: " + noAbsen + "\nNama\t: " +
nama
                + "\nNIM\t: " + nim + "\nAbsensi\t: " + dataHadir
+ "\n");

System.out.println("=========================================="
);
    }

    public boolean isNim(String nim) {
        return this.nim.equals(nim);
    }
}

class Absensi {
    LinkedList mahasiswa = new LinkedList();
    Scanner input = new Scanner(System.in);

    public boolean addOrang(int noAbsen) {
        String nama, nim;
        int hadir;

System.out.println("\n=========================================="
);
        System.out.print("Input Nama : ");
        nama = input.nextLine();
        System.out.print("Input NIM : ");
        nim = input.next();
        System.out.print("Pilih
:\n1.Hadir\n2.Sakit\n3.Alpa\nPilih Absensi : ");
        hadir = input.nextInt();
        input.nextLine();

System.out.println("==========================================");
```

```
        absenMahasiswa mahasiswa = new absenMahasiswa(nama, nim,
hadir, noAbsen);
        if (cekData(nim) == false) {
            this.mahasiswa.add(mahasiswa);
            System.out.println("Absen Berhasil!");
            return true;
        } else {
            System.out.println("Data yang diinput telah ada!");
            return false;
        }

    }

    public boolean cekData(String nim) {
        for (int i = 0; i < mahasiswa.size(); i++) {
            absenMahasiswa mhs = (absenMahasiswa)
mahasiswa.get(i);
            if (mhs.isNim(nim)) {
                return true;
            }
        }
        return false;
    }

    public void cariMahasiswa(int noAbsen) {
        for (int i = 0; i < mahasiswa.size(); i++) {
            absenMahasiswa m = (absenMahasiswa) mahasiswa.get(i);
            if (noAbsen == m.getNoAbsen()) {
                m.print();
            }
        }
    }

    public void remove() {
        mahasiswa.removeLast();
    }

    public void display() {
        for (int i = 0; i < mahasiswa.size(); i++) {
            absenMahasiswa m = (absenMahasiswa) mahasiswa.get(i);
            m.print();
        }
    }
}
public class BukuAbsensiMahasiswa {
    public static void main(String[] args) {
        Absensi ab = new Absensi();
        Scanner input = new Scanner(System.in);
        int cs, noAbsen = 1;
```

```
        boolean ulang = true;
        System.out.println("============ SELAMAT DATANG DI
APLIKASI ============\n"
                + "================= ABSENSI MAHASISWA
================");
        while (ulang) {
            System.out.print("\nPilih Operasi :\n1.Tambah Data"
                    + "\n2.Cari Mahasiswa\n3.Hapus Data Terakhir"
                    + "\n4.View Data\n5.Exit\nInput : ");
            cs = input.nextInt();
            System.out.println("");
            switch (cs) {
                case 1:
                    System.out.print("============== TAMBAH DATA
===============");
                    if (ab.addOrang(noAbsen)) {
                        noAbsen++;
                    } else {
                        break;
                    }
                    System.out.println("============= DATA
DITAMBAH ==============\n");
                    break;
                case 2:
                    System.out.println("============== CARI
MAHASISWA ===============");
                    System.out.print("Input nomor absen Mahasiswa
: ");
                    int noAbs = input.nextInt();
                    ab.cariMahasiswa(noAbs);
                    System.out.println("============ DATA TELAH
TAMPIL ===============");
                    break;
                case 3:
                    ab.remove();
                    System.out.println("=========== DATA BERHASIL
DIHAPUS ===========");
                    noAbsen--;
                    break;
                case 4:
                    System.out.println("================ VIEW
DATA =================");
                    ab.display();
                    System.out.println("============== AKHIR VIEW
==================");
                    break;
                case 5:
                    ulang = false;
                    break;
```

```
                default:
                    System.out.println("Inputan Salah!!");
                    break;
            }
        }
        System.out.println("Terima Kasih Banyak!");
    }
}
```

Hasil :

```
============ SELAMAT DATANG DI APLIKASI ============
================= ABSENSI MAHASISWA ================

Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 1

============== TAMBAH DATA ===============
==========================================
Input Nama : rizqillah
Input NIM : 1957301020
Pilih :
1.Hadir
2.Sakit
3.Alpa
Pilih Absensi : 1
==========================================
Absen Berhasil!
============= DATA DITAMBAH ==============

Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 1

============== TAMBAH DATA ===============
==========================================
Input Nama : rizqillah
Input NIM : 1957301020
Pilih :
1.Hadir
2.Sakit
3.Alpa
Pilih Absensi : 1
==========================================
Data yang diinput telah ada!
```

```
 Pilih Operasi :
 1.Tambah Data
 2.Cari Mahasiswa
 3.Hapus Data Terakhir
 4.View Data
 5.Exit
 Input : 1

 ============== TAMBAH DATA ===============
 ==========================================
 Input Nama : rizky
 Input NIM : 1957301063
 Pilih :
 1.Hadir
 2.Sakit
 3.Alpa
 Pilih Absensi : 1
 ==========================================
 Absen Berhasil!
 ============== DATA DITAMBAH ==============
Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 2

============== CARI MAHASISWA ===============
Input nomor absen Mahasiswa : 1
==========================================

No      : 1
Nama    : rizqillah
NIM     : 1957301020
Absensi : Hadir

==========================================
============ DATA TELAH TAMPIL ===============
```

```
Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 4

================ VIEW DATA =================
============================================

No      : 1
Nama    : rizqillah
NIM     : 1957301020
Absensi : Hadir

============================================
============================================

No      : 2
Nama    : rizky
NIM     : 1957301063
Absensi : Hadir

============================================
============== AKHIR VIEW ==================
Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 3

=========== DATA BERHASIL DIHAPUS ===========
```

```
Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 4

================ VIEW DATA ================
===========================================

No      : 1
Nama    : rizqillah
NIM     : 1957301020
Absensi : Hadir

===========================================
============== AKHIR VIEW =================
Pilih Operasi :
1.Tambah Data
2.Cari Mahasiswa
3.Hapus Data Terakhir
4.View Data
5.Exit
Input : 5

Terima Kasih Banyak!
BUILD SUCCESSFUL (total time: 4 minutes 5 seconds)
```

Penjelasan :

Pada program diatas ada beberapa fungsi yang telah digunakan. Dan maksud dari program tersebut adalah untuk menyimpan data-data absensi mahasiswa, yaitu di program tersebut telah menyediakan 5 fitur yang bisa dipake untuk pengolahan data-data yang ada. Yaitu ada fitur untuk menambah data, cari mahasiswa, hapus data terakhir, view data dan exit.

Fitur Tambah Data berfungsi untuk menambah data dari seorang mahasiswa, yaitu program akan meminta inputan nama mahasiswa, nim, dan absensi mahasiswa tersebut, pada absensi terdapat pilihan apakah mahasiswa tersebut hadir ataukah sakit ataupun alpa. Dan jika yang diinput adalah data dengan nim yang sama, maka akan ada pemberitahuan bahwa data telah ada.

Fitur Cari Mahasiswa berfungsi untuk mencari mahasiswa yang ingin dilihat datanya, dan pencarian mengambil data berdasarkan no absen dari mahasiswa tersebut. No absen didapati dari penginputan data-data mahasiswa, yaitu mahasiswa yang pertama diinput akan mendapat nomor absen pertama dan

seterusnya. Dan jika ada penghapusan data mahasiswa, maka nomor absen juga akan ikut terkurang atau decrement. Dan jika terjadi penambahan data, maka nomor absen akan ikut bertambah atau increment.

Fitur Hapus Data Terakhir berfungsi untuk menghapus data mahasiswa yang terakhir diinputkan. Dan ketika penghapusan terjadi, maka nomor absen juga akan ikut berkurang, dikarenakan jika mahasiswa yang terakhir diinputkan bernomor absen 3 maka jika data tersebut telah dihapus, maka nomor absen setelah penambahan data tersebut akan menjadi 4. Dan dikarenakan data mahasiswa tersebut dihapus. Maka nomor absen akan berkurang kembali ke 3.

Fitur View Data berfungsi untuk menampilkan seluruh data mahasiswa yang telah diinputkan beserta dengan informasi mahasiswa tersebut. Berupa nomor absen, nama, nim, dan keterangan absensinya.

Fitur Exit berfungsi untuk keluar dari perulangan program, dan jika keluar dari perulangan tersebut, maka akan membuat program selesai. Dan diwaktu selesai keluar dari loop, program akaan mencetak kalimat “Terima Kasih Banyak!”.

Adapun cara penyimpanan data sementara pada program tersebut menggunakan suatu class dari Java Collection yang bernama LinkedList. Alasan mengapa menggunakan LinkedList adalah dikarenakan pada program tersebut benyak menggunakan fungsi penghapusan dan penambahan data. Oleh karenanya program tersebut menyimpan data dalam LinkedList.

Analisis :

```
public class BukuAbsensiMahasiswa {
    public static void main(String[] args) {
```

➔ **Membuat class dari file java dengan nama BukuAbsensiMahasiswa**

```
        Absensi ab = new Absensi();
        Scanner input = new Scanner(System.in);
        int cs, noAbsen = 1;
        boolean ulang = true;
```

➔ **Membuat objek dari class Absensi, dan Scanner. Dan beberapa variabel lainnya beserta nilai awal.**

```
        System.out.println("============    SELAMAT    DATANG    DI
APLIKASI ============\n"
                +     "=================     ABSENSI     MAHASISWA
================");
        while (ulang) {
```

```
            System.out.print("\nPilih Operasi :\n1.Tambah Data"
                    + "\n2.Cari Mahasiswa\n3.Hapus Data Terakhir"
                    + "\n4.View Data\n5.Exit\nInput : ");
            cs = input.nextInt();
            System.out.println("");
```

➔ **Melakukan perulangan selama nilai ulang = true. Dan mencetak menu kelayar dan meminta inputan user menu yang akan dipilih.**

```
switch (cs) {
     case 1:
          System.out.print("==============       TAMBAH       DATA
===============");
          if (ab.addOrang(noAbsen)) {
               noAbsen++;
          } else {
               break;
          }
          System.out.println("=============     DATA     DITAMBAH
==============\n");
          noAbsen++;
          break;
```

➔ **Mengecek jika nilai inputan variabel cs = 1. Maka akan memanggil method untuk menambahkan mahasiswa melalui objek ab yang telah dibuat sebelumnya. Dan mengisi nilai parameter dengan nilai dari variabel noAbsen. Dan jika bernilai true kembalian, maka akan meng-increment nilai noAbsen. Dan jika bernilai false, maka akan break dari switch.**

```
class Absensi {
    LinkedList mahasiswa = new LinkedList();
    Scanner input = new Scanner(System.in);
```

➔ **Membuat class Absensi. Dan membuat objek dari LinkedList Java Collection dengan nama mahasiswa. Dan objek dari Scanner yang digunakan sebagai inputan data.**

```
public void addOrang(int noAbsen) {
        String nama, nim;
        int hadir;
```

➔ **Membuat method addOrang yang menerima parameter noAbsen. Dan membuat beberapa variabel yang dibutuhkan dalam method ini untuk menambah data mahasiswa nantinya.**

```
System.out.println("\n=========================================="
);
        System.out.print("Input Nama : ");
        nama = input.nextLine();
        System.out.print("Input NIM : ");
        nim = input.next();
        System.out.print("Pilih
:\n1.Hadir\n2.Sakit\n3.Alpa\nPilih Absensi : ");
        hadir = input.nextInt();
        input.nextLine();
System.out.println("==========================================");
```

➔ **Meminta beberapa inputan user untuk data dari mahasiswa. Yaitu inputan nama, nim dan pilihan apakah mahasiswa tersebut hadir atau sakit ataupun alpa. Dan dikarenakan disini inputannya beda-beda variabel, oleh karenanya digunakan pemanggilan method nextLine diakhir agar nantinya ketika meminta inputan data lagi, nextLine pada input nama tidak akan terlewati. Hal ini terjadi dikarenakan pengabaian enter setelah inputan nilai integer.**

```
      absenMahasiswa mahasiswa = new absenMahasiswa(nama, nim,
hadir, noAbsen);
      if (cekData(nim) == false) {
          this.mahasiswa.add(mahasiswa);
          System.out.println("Absen Berhasil!");
          return true;
      } else {
          System.out.println("Data yang diinput telah ada!");
          return false;
      }
```

➔ **Membuat objek dari class absenMahasiswa dengan nama mahasiswa, dan mengisi nilai parameter pada method construct dari variabel nama, nim, keterangan hadir, dan noAbsen. Kemudian method cekData yang diisi parameter nilai nim, dan dicek apakah kembalian method cekData sama dengan false, jika iya maka mengisi nilai kedalam objek mahasiswa. Dan jika tidak maka akan mencetak bahwa data telah ada.**

```
public boolean cekData(String nim) {
    for (int i = 0; i < mahasiswa.size(); i++) {
         absenMahasiswa mhs = (absenMahasiswa) mahasiswa.get(i);
         if (mhs.isNim(nim)) {
             return true;
```

```
        }
    }
    return false;
}
```

➔ **Melakukan pengecekan terhadap nilai nim yang diinput, dan melakukan perulangan sebanyak data dari linkedlist mahasiswa. Dan jika ditemukan nilai nim sama seperti nilai nim yang diinputkan, maka akan mengembalikan true, yang berarti datanya sudah ada. Dan jika selama perulangan tidak ditemukan sama, maka berarti data belum ada.**

```
class absenMahasiswa {
    private String nama, nim;
    private int hadir, noAbsen;
```

➔ **Membuat class absenMahasiswa. Dan mengatur beberapa variabel(properti) dari class, yaitu String nama dan nim. Dan variabel int hadir dan noAbsen.**

```
public absenMahasiswa(String nama, String nim, int hadir, int
noAbsen) {
        setNama(nama);
        setNim(nim);
        setHadir(hadir);
        setNoAbsen(noAbsen);
}
```

➔ **Program diatas adalah method construct dari class absenMahasiswa yang menerima beberapa paremeter, yang String nama, nim, dan int hadir dan noAbsen. Dan nilai yang diterima tersebut akan diinput kedalam variabel global pada class tersebut. Melalui method Setter.**

```
case 2:
    System.out.println("==============       CARI       MAHASISWA
===============");
    System.out.print("Input nomor absen Mahasiswa : ");
    int noAbs = input.nextInt();
    ab.cariMahasiswa(noAbs);
    System.out.println("============     DATA     TELAH     TAMPIL
===============");
    break;
```

➔ **Dan jika yang diinput user menu ke-2. Maka akan meminta input nomor absen mahasiswa yang akan dicari. Dan memanggil method**

cariMahasiswa melalui objek ab dengan mengisi nilai parameter dari nomor absen yang akan dicari.

```
public void cariMahasiswa(int noAbsen) {
   for (int i = 0; i < mahasiswa.size(); i++) {
       absenMahasiswa m = (absenMahasiswa) mahasiswa.get(i);
       if (noAbsen == m.getNoAbsen()) {
           m.print();
       }
   }
}
```

➔ **Membuat method untuk mencari mahasiswa menurut noAbsen. Dan melakukan perulangan selama nilai i=0, dan i< dari jumlah data pada objek mahasiswa. Maka akan membuat objek dari class absenMahasiswa bernama m, dan diisi dari nilai data objek mahasiswa pada index ke-i. Dan mengecek apakah nilai noAbsen sama dengan nilai pada method getNoAbsen melalui objek m. Jika sama maka akan memanggil method print melalui objek m agar mencetak data yang dicari.**

```
public void print() {
    String dataHadir;
    if (hadir == 1) {
        dataHadir = "Hadir";
    } else if (hadir == 2) {
        dataHadir = "Sakit";
    } else {
        dataHadir = "Alpa";
    }

System.out.println("=========================================="
);
    System.out.println("\nNo : " + noAbsen + "\nNama : " + nama
         + "\nNIM : " + nim + "\nAbsensi : " + dataHadir + "\n");

System.out.println("=========================================="
);
}
```

➔ **Membuat method print dalam class absenMahasiswa. Dan membuat variabel dataHadir. Dan mengecek apakah nilai hadir sama dengan 1 maka akan mengisi dataHadir menjadi “Hadir” dan jika hadir sama dengan 2 maka akan mengisi dataHadir menjadi “Sakit” dan jika hadir sama dengan 3 maka akan mengisi**

dataHadir menjadi "Alpa". Kemudian mencetak data mahasiswa kelayar.

```
case 3:
    ab.remove();
    System.out.println("===========    DATA    BERHASIL    DIHAPUS
==========");
    noAbsen--;
    break;
```

➔ **Mengeksekusi statement case 3 jika pilihan menu adalah 3. Dan memanggil method remove melalui objek ab. Dan mengurangi nilai noAbse dengan decrement.**

```
public void remove() {
   mahasiswa.removeLast();
}
```

➔ **Membuat method remove yang digunakan sebagai untuk menghapus data terakhir yang diinputkan. Pada method ini menggunakan pemanggilan method removeLast dari objek mahasiswa.**

```
case 4:
    System.out.println("================         VIEW         DATA
=================");
    ab.display();
    System.out.println("==============         AKHIR         VIEW
==================");
    break;
```

➔ **Jika pilihan yang diinput pada menu adalah 4, maka akan melakukan statement berikut. Yaitu akan memanggil method display melalui objek ab.**

```
public void display() {
    for (int i = 0; i < mahasiswa.size(); i++) {
         absenMahasiswa m = (absenMahasiswa) mahasiswa.get(i);
         m.print();
    }
}
```

➔ **Membuat method display yang digunakan sebagai untuk menampilkan seluruh data dari objek mahasiswa. Pertama adalah melakukan perulangan selama nilai i = 0 dan i < dari nilai jumlah data mahasiswa. Dan membuat objek dari class absenMahasiswa bernama m dan diisi dengan data dari pengambilan get pada index ke-i melalui objek mahasiswa. Dan**

memanggil method print pada class absenMahasiswa melalui objek m.

```
case 5:
    ulang = false;
    break;
```

➔ **Jika yang diinputkan adalah 5 pada pilihan menu. Maka akan membuat nilai variabel ulang yang bertipe boolean menjadi false. Oleh karenanya maka akan keluar dari perulangan yang ada.**

```
default:
     System.out.println("Inputan Salah!!");
     break;
```

➔ **Jika yang bukan angka antara 1 dan 5, maka akan menampilkan bahwa inputan salah!.**

```
System.out.println("Terima Kasih Banyak!");
```

➔ **Dan jika sudah keluar dari perulangan, maka akan mencetak kalimat "Terima Kasih Banyak!".**

```
public boolean isNim(String nim) {
    return this.nim.equals(nim);
}
```

➔ **Method diatas berguna untuk melakukan pengecekan terhadap nilai nim yang diinput dengan nilai nim pada variabel nim. Agar diketahui apakah nilai nim yang diinput sama seperti nilai pada data.**

BAB III
PENUTUP

3.1 Kesimpulan

Dapat disimpulkan, bahwa dalam membuat sebuah program, konsep OOP(Object Oriented Programming) sangat dibutuhkan, yaitu agar program yang dibuat bisa menjadi lebih simple, elegan, mudah dimodifikasi, dan lain-lain.

Untuk membuat program agar bisa dikatakan telah mempunyai konsep OOP adalah dengan adanya beberapa dari konsep Class, Object, attribute, Method, Construct, Inheritance, Polymorphism, Encapsulation, dan Abstract. Pada pembuatan suatu program yang mungkin membutuhkan hal yang sangat kompleks. Maka konsep OOP tersebut sangatlah penting dipelajari. Dan dengan adanya OOP pada pemrograman, membuat suatu program menjadi lebih mudah dan powerful.

Interface List merupakan sub-interface dari interface Collection. Interface List digunakan untuk mengkoleksi data dalam bentuk terurut dan memperbolehkan duplikasi. Interface List menambahkan operasi yang berkaitan dengan posisi.

ArrayList menyimpan data seperti array (diakses dengan index) namun ukurannya dapat bertambah secara fleksibel, dikarenakan ketika data yang dimasukkan melebihi ukuran array yang sekarang, maka arraylist akan membuat array baru yang dapat menampung elemen yang lebih banyak. Elemen yang dapat dimasukkan dalam ArrayList bisa bermacam-macam, termasuk null.

LinkedList adalah bagian dari Java Collection yang ada dalam paket java.util. Kelas ini merupakan implementasi dari struktur data LinkedList yang merupakan struktur data linier dimana elemen tidak disimpan di lokasi yang berdekatan dan setiap elemen merupakan objek terpisah dengan bagian data dan bagian alamat. Elemen-elemen tersebut dihubungkan menggunakan pointer dan alamat.

DAFTAR PUSTAKA

[1] Diki Alfarabi Hadi. "PHP OOP". https://www.malasngoding.com/php-oop-part-2-pengertian-class-object-property-dan-method. MalasNgoding.

[2] Andre. "Tutorial Belajar OOP PHP". 29 September 2014. https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-class-object-property-dan-method/. Dunia Ilkom.

[3] Rully Febrian. "OOP: Memahami Object Class Properti dan Method". 14 Juli 2014. https://www.dumetschool.com/blog/OOP-Memahami-Object-Class-Properti-dan-Method. DumetSchool.

[4] Viska Mutiawani dan Kurnia Saputra. 11 Maret 2014. http://informatika.unsyiah.ac.id/. Informatika Unsyiah.

[5] Oracle. "Class LinkedList". https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

[6] GeeksForGeeks. "LinkedList in Java". 05 Maret 2021. https://www.geeksforgeeks.org/

[7] javaTpoint. "Java LinkedList Class". https://www.javatpoint.com/

[8] W3Schools. https://www.w3schools.com/java/java_linkedlist.asp.

[9] Appkey. "Perbedaan ArrayList dan LinkedList". https://appkey.id/pembuatan-aplikasi/aplikasi-android/java-class/. 28 Januari 2021.

[10] GeeksForGeeks. "ArrayList in Java". 26 Juni 2020. https://www.geeksforgeeks.org/arraylist-in-java/

[11] javaTpoint. "Java ArrayList Class". https://www.javatpoint.com/java-arraylist

[12] QAStack. "HashSet vs TreeSet". https://qastack.id/programming/1463284/hashset-vs-treeset

[13] javaTpoint. "Java HashSet Class". https://www.javatpoint.com/java-hashset

[14] GeeksForGeeks. "HashSet in Java". 09 September 2020. https://www.geeksforgeeks.org/hashset-in-java/

[15] javaTpoint. "Java TreeSet Class". https://www.javatpoint.com/java-treeset

[16] GeeksForGeeks. "TreeSet in Java with Example". 26 Juni 2020. https://www.geeksforgeeks.org/treeset-in-java-with-examples/

[17] Oracle. "Class TreeSet". https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html.

[18] Oracle. "Class ArrayList". https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

[19] Oracle. "Class HashSet". https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html